# 1
# Introduction

## 1.1
## Context and Motivation

### Real Scenario

The growth of the Internet has exposed applications to thousands of concurrent users, who are served by means of sophisticated computing infrastructures. A *data center* is a facility that houses computer clusters and their associated components, such as storage and telecommunications subsystems. There are many examples of business that rely on data centers and a single hour of downtime of those services such as Google, MSN, and Yahoo! could often result in millions of dollars of lost revenue, bad publicity, and 'click over' to competitors.

The complexity of data centers poses many challenges for system administrators, which must constantly monitor their infrastructure to ensure they are providing appropriate levels of service to their users. Supervisory processes are fundamental when running large systems and critical operations that needs to be resilient to faults. It is desirable *to devise an automaton that can detect anomalies* as soon as possible in the infrastructure and raise informative alarms so that human experts can check the system and promptly act when needed and are spared from the rather implausible burden of continuous monitoring.

Every server, application and network device exports statistics that can be collected and monitored. Metrics are usually collected every minute so as to reduce load on the network, but some operations may require higher resolution[1]. Operators have long relied upon monitoring software to analyse the current state of the infrastructure, nevertheless it is still very difficult to effectively monitor an entire data center. Traditional monitoring software, such as Cricket and Nagios, requires the *manual configuration* of thresholds for every data stream to be monitored: an expert spends significant amount

[1]For example, in Twitter metrics are collected every 10 seconds according to Velocity Conference 2009 video available at `http://velocityconference.blip.tv/file/2300327/`

of time to define what is considered normal behavior for each data stream using a constant threshold value. This strategy is very poor and does not take advantage of all data available, specially because several measurements from distinct sources are strongly correlated – for example, redundant network links sharing traffic and different machines running same applications with load balancing. Furthermore, many alerts associated with individual streams may be symptomatic of the same *failure* and with the advent of hardware virtualization[2], the manual configuration of these threshold becomes even more impractical and the need for better tools is evident.

This work is motivated by the real need to improve the current monitoring solution at the data center of one of the largest Internet Service Provider (ISP) in South America – it has over 1,200 servers generating roughly 40,000 charts which operators need to browse in order to explore the current state of the infrastructure. A more intelligent monitoring system should decrease the time to detect faults, less the burden on the operations team and provide additional contextual information when an anomaly occurs to make troubleshooting more efficient.



## Data Streams

In this context, massive amounts of data are produced and it is not usually feasible to store all the data. In fact, most data centers only store a *fixed* amount of average monitoring data[3].

The streaming data model may be viewed as a generalisation of the traditional data warehouse model when the dataset size grows to infinity. In the most general case, each individual data record may be a tuple of arbitrary data types (e.g., binary, numerical, or string). This work focus on *numerical* values since they are ubiquitous to all monitored entities, hence a stream is considered to be a semi-infinite multivariate time series. Thus, for our investigation, we consider the following multiple stream data model.

**Definition 1** (Stream data model) *The dataset* $\mathcal{Z}$ *is a growing sequence of* $N$*-dimensional vectors, i.e.* $\mathcal{Z} \equiv \{\boldsymbol{z}(1), \boldsymbol{z}(2), \ldots, \boldsymbol{z}(t), \boldsymbol{z}(t+1), \ldots\}$ *with* $\boldsymbol{z}(t) \in \Re^N$ *for* $t \geq 1$.

In practice, there are $N$ single streams being generated asynchronously from possibly different sources and need to be combined to form the derived

[2]Virtual machines are used to abstract the physical infrastructure and resources can be dynamically allocated proportional to demand.

[3]*Round Robin Database* is the standard technique used in the industry, where it's common to retain only the current day's monitoring data with a good resolution.

multidimensional signal of interest. The task of gathering data imposes challenges by itself and this work assumes it is necessary to have all measurements available together, eventually, in a single snapshot at uniform time intervals for the actual processing – thus, distributed analysis is not in the scope of this work.

Researchers have started to redesign traditional mining algorithms to satisfy the requirements of the data streams context, where a streaming algorithm must be incremental and work with one or few passes over the data. In the past few years, a new theory has emerged for reasoning about algorithms that work within these constraints on space, time and number of passes. Some of the methods rely on pseudo-random computations, sparse approximation theory and synopsis structures – such wavelets, sketches and histograms [see Aggarwal, 2007, Gama and Gaber, 2007, Muthukrishnan, 2005].

There is also a lot of work on Data Stream Management Systems, which include Aurora [Abadi et al., 2003], Stream [Arasu et al., 2004], Telegraph [Chandrasekaran et al., 2003], Gigascope [Cranor et al., 2003] as well as recent implementations from the industry such as Esper, Corel8 and Apama. The common goal of these systems is to allow querying and joining streams over window sizes and computing global aggregate statistics, but do not focus on finding patterns in data or detecting anomalies.

There are clear opportunities to combine machine learning algorithms with DSMSs and leverage the pre-processing of the incoming raw data.



## Anomaly Detection

Anomaly detection is an important and challenging problem that has been treated within diverse domains and research areas, such as *statistics, machine learning data mining, information theory, spectral theory* and refers to the problem of finding patterns in data that do not conform to expected behavior. These non-conforming patterns are often referred to as *anomalies, outliers, change detection, exceptions, discords, aberrations, surprises, novelties, peculiarities or contaminants* in different application domains.

In data centers, an anomaly is a sudden and short-lived deviation from the normal operation of the network. Some anomalies can be the result of malicious actions (e.g. denial-of-service attacks) but they are mostly due to faults in the interaction between components in the system, specially under heavy load; common source of anomalies are: software bugs (e.g. memory leaks, not optimized database queries), hardware malfunctioning (e.g. disk failures) or faults in the underlying subsystems (e.g. broken communication network

access link).

Operations teams need to be informed in real-time when a problem occurs so it can be analysed and the appropriate action can be taken. Therefore, it is not essential to detect the entire anomalous subsequence and we can simplify the problem to only detect the instant an anomaly starts. Given the lack of universal definition or underlying framework to discuss anomaly detection, we propose the following definition for the problem of detecting anomalies in data streams.

**Definition 2** (Anomaly detection) *An anomaly $\alpha_i$ that starts at instant $\tau_i$ and lasts for $\ell_i$ intervals is represented as $\alpha_i = (\tau_i, \ell_i)$. Given the data so far, the task is to output a detection at interval $d_i$ for every $\alpha_i$ such that $\tau_i \leq d_i \leq \tau_i + \ell_i$ and $(d_j - \tau_i)$ is minimal.*

The performance of a solution to the task is evaluated in terms of the precision of detections considering the false alarms and missed anomalies. In real datasets, less than 5% of the data points correspond to anomalies, therefore the number of true negatives is very high and consequently the false positive rate is not a very meaningful measure. Therefore, we consider the precision and recall measures as they give a more informative picture, and use the usual $F_1$ score to represent both metrics.

There are three fundamental approaches to the problem [Hodge and Austin, 2004]:

**Supervised** techniques model both normality and abnormality and require pre-labelled data. This is the traditional approach, where a classifier can be trained with these samples.

**Semi-supervised** techniques model only normality and are more applicable than the previous approach since only labels for normal data is required.

**Unsupervised** techniques require no prior knowledge of the data, but must assume that anomalies are somehow distant from normal data.

The recent surveys [Chandola et al., 2009, Patcha and Park, 2007] on the subject note that anomaly detection has traditionally dealt with record or transaction type data sets. They further indicate that most techniques require the entire test data before detecting anomalies, and it mentions very few online techniques. Indeed, most current algorithms assume the dataset fits in main memory [Yankov et al., 2008]. Both aspects violate the requirement for real-time monitoring data streams. Regardless of this restriction, the vast majority of techniques closer to our domain consider specific modelling techniques

based on attributes from network packet traces, such as packet counts, byte counts, and number of open connections [Gu et al., 2005, Lakhina et al., 2005, 2004a, Barford et al., 2002]. Moreover, most approaches focused specifically on intrusion detection [Kuang and Zulkernine, 2008, Xu et al., 2005, Lee and Stolfo, 2000]. From a database point of view, most existing research on data streaming computation focus on heavy-hitter detection; however, heavy-hitters do not necessarily correspond to anomalies. Schweller et al. [2004] uses reversible sketches for heavy change detection, but it is still specific for IP header data.

A comparative study [Chandola et al., 2008] between methods for detecting anomalies in symbolic data shows there are several techniques for obtaining a symbolic representation from a time series [for instance Lin et al., 2007, Bandt and Pompe, 2002], but all works seem to solely apply to univariate data [such as in Keogh et al., 2004, Wei et al., 2005].

It's a challenging task to detect failures in large dynamic systems because anomalous events may appear rarely and do not have fixed signature. The high dimensionality of the observation data due to the complexity of the systems, together with the diversity of data sources and the frequent changes of system normal behavior resulting from user behaviour makes detection even more difficult. Therefore, one cannot assume a stationary process and valid approaches must account for concept drifts.

In order to make the technique most widely applicable, an any-time *unsupervised* method that does not require training data is greatly desired as the nature of normal measurements can be learnt and it can autonomously adapt to variations in the structure of 'normality'. Because it is difficult to obtain training data, and we are more concerned with problems never seen before that notoriously arise in complex systems.

Despite the fact that outlier detection has been studied since the $19^{th}$ century [Edgeworth, 1887], the problem is still open in the sense that not many techniques addresses all the presented requirements: streaming, unsupervised, multivariate and non-stationary data.

## 1.2 Objectives

This work aims to study unsupervised techniques to detect anomalies in multivariate time series under the streaming constraints with the further objective to implement a real-time solution to support operations teams overseeing a data center. Hence, the work focus on measurements that are collected from data centers and naturally exhibit local correlations (such as

| Symbol | Description |
|---|---|
| N | Number of streams. |
| $\boldsymbol{z}(t)$ | Data snapshot $\boldsymbol{z}(t) \equiv [z_1(t) \dots z_N(t)]^\top$ at time $t$. |
| $\alpha$ | Forgetting factor. |
| $\boldsymbol{\Phi}(t)$ | Incrementally estimated covariance $N \times N$ matrix. |
| $r$ | Number of latent variables. |
| $\boldsymbol{Q}(t)$ | $N \times r$ projection matrix. The column vectors are the principal subspace basis. |
| $\boldsymbol{h}(t)$ | $r \times 1$ vector of latent variables. |
| $\tilde{\boldsymbol{z}}$ | $N \times r$ vector of the reconstruction of $\boldsymbol{z}(t)$. |
| $E_t$ | Energy up to time $t$. |
| $\tilde{E}_t$ | Energy captured by the latent model up to time $t$. |
| $f_E, F_E$ | Lower and upper bounds on the fraction of total captured energy. |
| $T$ | Total number of intervals so far. |

Table 1.1: Description of notation.

measurements from machines in the same cluster and from redundant network links).

Anomaly detection involves extracting relevant information from high-dimensional, high-rate, noisy data. As part of this study, we review the current literature and in particular, we highlight two prominent approaches for the anomaly detection problem: one based on principal subspace tracking [Hoke et al., 2006] and another based on kernel-RLS to identify a high-density region in space [Ahmed et al., 2007a]. The different methods are to be implemented so that they can be empirically evaluated on the public datasets used in the literature as well as on the datasets from our target ISP.

## 1.3 Notation

## 1.4 Contributions

Our work points out that SPIRIT [Papadimitriou et al., 2005], the subspace tracker algorithm used for anomaly detection in InteMon [Hoke et al., 2006], is an extension to the PASTd algorithm [Yang, 1995a] and thus inherits two major disadvantages that are widely known in the signal processing community: inability to provide orthonormal estimates and instability in the updating of the inverse power matrix that will 'explode' in the case of fading signals due to its use of the matrix inversion lemma. The loss of orthonormality in the traked basis is critical because it is necessary by the rank estimation

procedure in order to keep the reconstruction error respecting the algorithm's parameters. Therefore, in practice it really requires $\mathcal{O}(Nr^2)$ to remain correct due to the extra orthonormalization step and not $\mathcal{O}(Nr)$ as advertised. However, there is no known remedy for the numerical instability vulnerability. We also observe that numerical instabilities plague the kernel-RLS based KOAD algorithm where inputs must be normalized otherwise the algorithm just fails.

We address those disadvantages and present a better algorithm for anomaly detection. Namely, we introduce FRAHST (Fast Rank-Adaptive row-Householder Subspace Tracker) which is the first extension to the state-of-the-art recursive row-householder subspace tracking algorithm introduced in Strobach [2009a] and adds the ability to also track the subspace rank $r$. The algorithm is an implementation of the orthogonal principle where the $Q$ and $S$ matrices are directly updated through a recursive Householder reduction. In short, FRAHST is to Strobach's algorithm as SPIRIT is to Yang's and we offer implementation details on how to reduce the computational complexity in solving the linear system in the algorithm from $\mathcal{O}(r^3)$ to $\mathcal{O}(r^2)$ by exploiting recurrent 'rank-one' updates of QR factors of a pair of internal matrices. Hence, our proposed approach provides excellent orthonormal estimates and is purely reflection based thus avoiding any inverse power matrix updating. The algorithm reaches a complexity of $\mathcal{O}(Nr)$ which is the lower bound for an algorithm of this kind.

We compare our method with four different related anomaly detection techniques, and FRAHST gives excellent results for the public Abilene dataset and provides the best overall results for our own dataset. It is also the only algorithm that consistently has a $F_1$ score equal or greater than 80% in all datasets. A scalable event-driven architecture is devised and a real-time system is implemented where a DSMS is used to correlate the incoming raw stream data and feed to FRAHST so alerts can be raised once a new sample arrives that cannot be explained in a satisfactory manner by the current model (i.e. indicated by a increase in $r$). The devised solution successfully promotes the integration between machine learning and DSMSs where good execution performance is shown to be achieved.

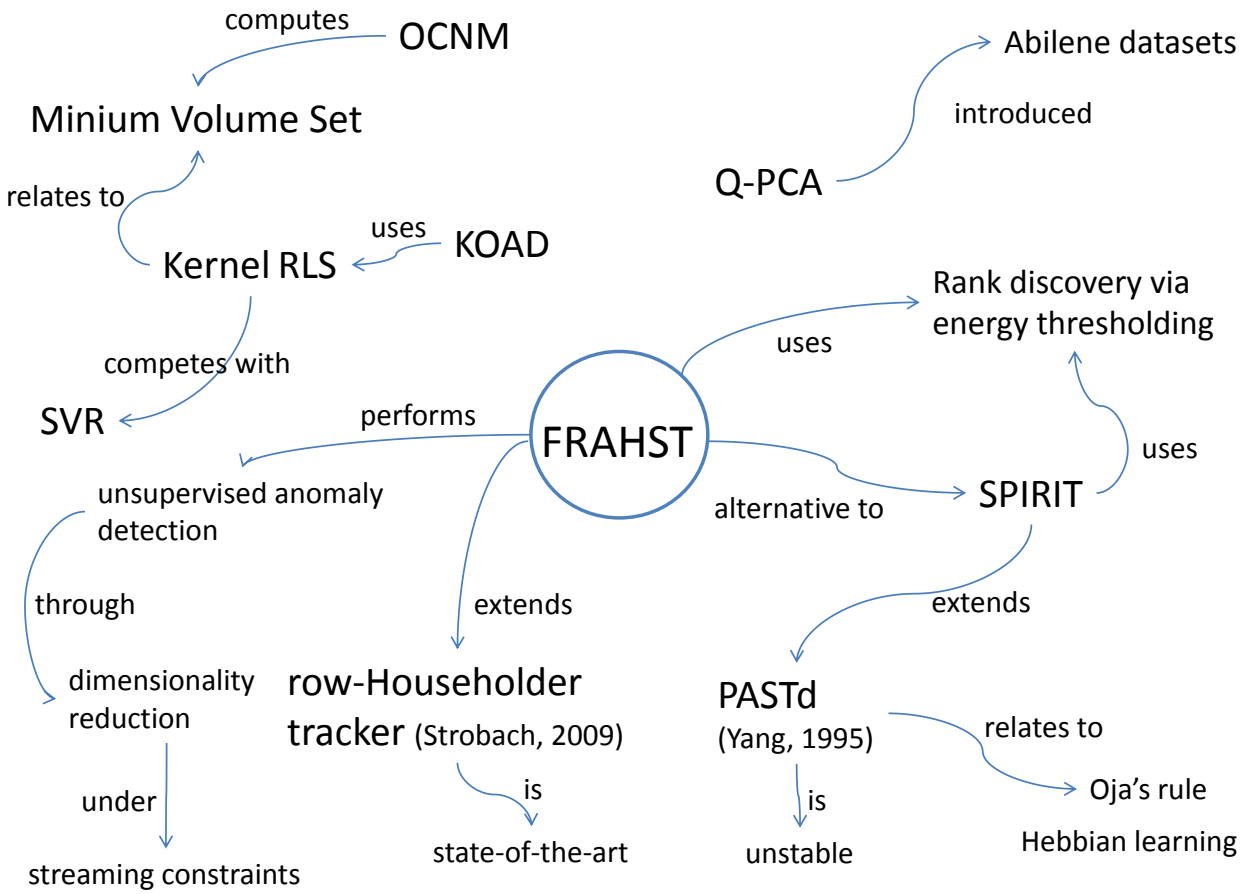


Figure 1.1: Overview of this work.

## 1.5 Dissertation structure

The dissertation is organized in the following way. Chapter 2 outlines the most relevant anomaly detection work. In chapter 3, the fundamental background against the subspace trackers provides an overview necessary to enable the narrative over the proposed FRAHST algorithm which is presented in chapter 4 where we formalize how the algorithm is used for anomaly detection and describe the implementation of a real-time data center monitoring system. Finally, experiments and results are shown in chapter 5 and chapter 6 concludes this dissertation with a discussion on the results and contributions and offer various possibilities for future work.